Q-LEARNING BASED CONTROL ALGORITHM FOR HTTP ADAPTIVE STREAMING

Virginia Martín, Julián Cabrera, and Narciso García

Abstract—**We present a control algorithm based on Q-Learning for an HTTP Adaptive Streaming (HAS) Client in order to optimize the Quality of Experience (QoE) of the user. First, we propose a model with a suitable number of variables in an attempt to find a reasonable tradeoff between the complexity of the model and its capacity to capture appropriately the dynamics of the system. Second, we define a novel reward function that takes into consideration factors related to the user's QoE. Results will show, that our Q-learning algorithm is able to learn and efficiently control the selection of the segment qualities. In addition, we will show that our proposed approach outperforms another Q-learning approach.**

Index Terms—**HTTP Adaptive Streaming, reinforcement learning, state, reward, policy, adaptation, Quality of Experience**

I. Introduction

The growth of video traffic evidences the importance of carrying on a good management of this multimedia traffic, including an efficient encoding, a smart distribution of that content optimizing the use of the network resources, among others aspects.

HTTP Adaptive Streaming (HAS) is becoming a widely adopted solution to deliver video streaming to end-users in highly dynamic networks. The goal of this technology is to adapt the bit-rate of the stream to the available network throughput in order to deliver the best possible video quality to the user at any given time, considering the bandwidth changes, the content characteristics and the device features. In all the HAS architectures, a video stream is encoded with different encoding parameters (bitrate, resolution, etc.) and the resulting streams, called representations, are split into segments that are stored in a server. The client requests each segment at the most appropriate quality level according to the system conditions using its particular adaptation algorithm. Therefore, HTTP Adaptive Streaming is a pull technology since the client is the intelligent entity and it has the session control exclusively by means of segment requests. Moreover it is based on an HTTP/TCP scheme taking advantage of some features such as, it does not suffer from NAT and firewall issues, the HTTP-delivery allow to use standard HTTP servers and caches, and TCP guarantees that the data are received in the client side and at the same order in which they were sent.

There are several proprietary HTTP Adaptive Streaming (HAS) solutions, as well as, an international standard, MPEG-DASH (Dynamic Adaptive Streaming over HTTP)[1], intended to support a media streaming model for the delivery of media content. Although, it defines several aspects of this streaming technology, such as, the format of the Media Presentation Description (MPD) file or the segment format, it does not specify the control algorithm for the selection of the quality of the requested segments.

Several approaches have been proposed to control the quality decision in a scenario with a single HAS client. The adaptation algorithm proposed by Miller et al. [2] is based on the control of the level of the buffer, being its main priority to avoid playback freezes. This heuristic algorithm chooses the lowest representation for the first segment with the aim of minimizing the initial delay. Then, immediately it increases the quality aggressively of the requested segments. After this phase, the algorithm tries to keep the buffer level close to an predefined optimum value B_{opt}. On the other hand, the algorithm proposed by Liu et al. [3] uses the segment fetch time, instead of the instantaneous TCP transmission rate, to detect the bandwidth changes. In case of a decision to switch up, the algorithm will select the next higher representation level with the goal of preventing playback freezes. On the contrary, in case of switching down, an aggressive quality downgrade is taken.

Other research studies model the problem as an optimization control problem and use mathematical tools to solve it. The solution proposed by García [4] is based on Stochastic Dynamic Programming (SDP). They compute off-line optimal control policies based on a system model that is estimated apriori. This scheme produces good results provided that the used models match the real behavior of the system. Nevertheless it lacks of adaptiveness if there is a mismatch between them.

Reinforcement learning techniques are also used to determine the quality of the next segment. In the self-learning client algorithm proposed by Claeys [5], each environment state is defined using six variables: the *buffer level*, the *available throughput*, the *previous requested quality*, the *buffer filling change*, the *oscillation length* and the *oscillation depth*. Although few variables can lead to a bad policy due to an incorrect modeling of the environment, too many variables can involve some convergence problems and a slow learning process. In [6], the authors enhance the previous system [5], reducing the number of variables, re-defining some functions that are key for the performance of the solution and combining

the Q-algorithm with eligibility traces in order to obtain a method that may learn more efficiently.

In this paper, we propose an adaptation algorithm based on Q-learning approach. The main contributions of this paper are two-fold. First, we propose a model with a suitable number of variables in an attempt to find a reasonable tradeoff between the complexity of the model and its capacity to capture appropriately the dynamics of the system. Second, we define a novel reward function that takes into consideration factors related to the user's QoE: the requested qualities, the quality switches, and the freezes. Results will show, that our Q-learning algorithm is able to learn and efficiently control the selection of the segment qualities. In addition, we will show that our proposed approach outperforms other Q-learning approaches, assessing our system characterization and proposed reward function.

The paper is organized as follows. Section II describes the proposed Q-Learning HAS algorithm. In Section III we evaluate its behavior using different exploration policies and its effectiveness compared with another self-learning client based on an alternative of Q-learning. Finally, in Section IV we present the conclusions of the work.

II. DASH-QL: Our proposed Q-Learning-based adaptation algorithm

A. Introduction

Q-learning [7], [8] is a **model-free reinforcement learning technique** that allows a dynamic system to learn which is the most appropriate action for the next stage depending on specific environment conditions at each time. The system is characterized by a set of possible states, and the algorithm has to determine the action to be taken according to the state it is in. This technique combines an exploration mode where random actions are selected and the algorithm learns from their outcome, and an exploitation mode in which the algorithm selects the most suitable action according to what it has learned up to that moment. Thus, the strengths of Q-learning are: no previous knowledge of the system dynamics is required, since the client only uses its previous experience; and the policy is not fixed and it could change progressively allowing a higher adaptation degree to the dynamic conditions.

B. Elements of the DASH-QL approach

The main elements of our Q-learning formulation of the control of the adaptive streaming client are:

- ***State:*** Contains relevant information about the environment conditions at each time. Specifically, in our model we define the state as follows: $s_k = (buf_k, bw_k, q_k)$ considering
  - The level of the client's buffer in seconds [*buf*]. A value of zero indicates that a freeze has occurred.
  - The available bandwidth in Kbps [*bw*]. We estimate this value from the download time of the previous segment.
  - The bitrate of the previous requested quality level in Kbps [*q(j)*], where *j* is the associated quality level.

  The values of bw_k (1) and buf_k (2) from state to state are computed as follows:

$$bw_k = \frac{q_k \cdot T_{seg}}{\Delta t_{k-1}} \quad (1) \qquad buf_k = buf_{k-1} + T_{seg} - d_k \cdot T_{seg} \quad (2)$$

  where T_{seg} is the segment duration and d_k is the number of segments extracted from the buffer during the download time, Δt_k.
- ***Action:*** The different available qualities of the segments.
- ***Reward function:*** The function that indicates how good the decision taken is. Our reward function is composed of three factors related to the quality of experience of the user, whose coefficients have been empirically selected:

$$R_{total} = R_{quality} + R_{switches} + R_{freezes} \quad (3)$$

  - The factor $R_{quality}$ aims at favoring the selection of higher qualities, but taking into account the buffer level. This term is calculated as shown in (4).

$$R_{quality} = -1.5 \cdot \left| bw_k \frac{1 + \left(\frac{buf_k}{Bopt}\right)}{2} - q(sel\_iq_k) \right| \quad (4)$$

    where the variable sel_iq_k identifies the quality level for the current download and the variable B_{opt} indicates the target optimal buffer value.
    When the buffer level is lower than B_{opt}, this term rewards the selection of lower bitrates in order to increase its level. Whereas, if the buffer is greater than B_{opt}, it favors the selection of higher qualities.
  - The factor $R_{switches}$ (5) penalizes the occurrence of quality switches between two consecutive requests. Although a quality change that entails a quality enhancement is assessed positively, a noticeable jump might penalize the QoE.

$$R_{switches} = -1 \cdot \left| q_k - q(sel\_iq_k) \right| \quad (5)$$

  - And lastly, $R_{freezes}$ is the penalization factor of the video freezes. As they harm seriously the perceived quality by the user, when this situation happens the client receives a high negative reward as shown in (6).

$$R_{freezes} = \begin{cases} -50000 & if\ underflow = 1 \\ 0 & otherwise \end{cases} \quad (6)$$

- ***Q-table,*** $Q(s,a)$***:*** The rows of this matrix are all the states of the system and each column contains one of the possible actions (the segment qualities). For a given pair (s,a), $Q(\cdot)$ indicates the learned benefit that the system will get taking action a in state s. In the exploitation mode, the algorithm will select the action with the highest $Q(\cdot)$ value for a given state, s. In order to include what the client has learned taking an action, the Q-Learning approach updates this matrix after each quality decision as follows [7]:

$$Q(s_k, a) \rightarrow (1-\alpha) \cdot Q(s_k, a) + \alpha \cdot \left[r + \gamma \cdot max_b Q(s_{k+1}, b) \right] \quad (7)$$

  where s_k is the current state, a is the selected action, r is the associated immediate reward, b is the action that produces the highest Q-value for the next state s_{k+1}, the *learning rate* (α) indicates how much the acquired information will affect

to the old value of $Q(\cdot)$ in its updating and the discount factor (γ) that weighs the contribution of the immediate and future rewards ($0 \leq \gamma \leq 1$).

III. Performance Comparison and Results

A. Experiments Setup

Each simulation has consisted of 120 episodes with 300 segments each one, so we ensure that the convergence is achieved. In this sense, the obtained results have been computed considering the last 20 episodes in order to assess the performance of the algorithms after the learning phase.

The simulations have been carried out using the following parameter values: the segment length is T_{seg} = 2 seconds and each segment has been encoded at $N_q = 14$ quality levels with bitrates Q_i distributed between 100 and 4500 Kbps. The buffer size is 20 seconds and we have empirically selected a value of 10 seconds for the B_{opt} parameter. The available channel throughput is modeled by a Markov chain with $N_{bw} = 15$ quantified levels and a remaining probability $p = 0.2$. Regarding the learning rate and the discount factor, we have taken the values that Claeys describes as the best configuration [6], $\alpha = 0.1$ and $\gamma = 0.1$ respectively. Moreover, we use a QoE measurement in order to evaluate the algorithm performance objectively. It is defined as follows [8]:

$$QoE = 4.85 \cdot Q - 4.95 \cdot F - 1.57 \cdot S + 0.5 \tag{8}$$

Where the factor Q [9] indicates the average segment quality and it affects positively to the QoE.

$$Q = \frac{1}{N \cdot q(N_q)} \cdot \sum_{j=1}^{N_q} q(j) \tag{9}$$

The factor F [10] depends on the frequency fr_{freq} and the length fr_{time} of the video freezes and it degrades the QoE.

$$F = \frac{7}{8} \cdot \left(\frac{ln(fr_{freq})}{6} + 1 \right) + \frac{1}{8} \left(\frac{min(fr_{time}, 15)}{15} \right) \tag{10}$$

Finally, the factor S [11] shows how the quality switches influence in the perceived quality, where sw_{number} and sw_{depth} represent the number of switches and their average bitrate depth, respectively.

$$S = \frac{sw_{number} \cdot sw_{depth}}{N \cdot (q(N_q) - q(1))} \tag{11}$$

B. Influence of the exploration policy

The strategy adopted to choose exploratory actions plays an important role on the algorithm performance and the convergence speed. The different exploration policies used are described below:

- The first method is $\epsilon-$greedy with a fixed value to $\epsilon = 0.1$. With this method, the probability of selecting is distributed uniformly among all the actions.
- The second one is the Value-Difference Based Exploration with Softmax action selection policy (VDBE-Softmax)[9]: the client selects random actions using the Softmax probabilities in case of that $\xi < \epsilon$ and it chooses the greedy action otherwise. The Softmax probabilities are determined through the Boltzmann distribution proposed by Tokic [9] using a normalization of the Q values into the interval [-1,0] and a value for the temperature parameter equals $\tau = 0.01$. With this technique, we use two definitions for ϵ:
  - A constant exploration rate with a value of $\epsilon = 0.1$.
  - An exploration rate $\epsilon(s)$ depending on the state that is updated according to [12] proposed by Claeys [6], using the mentioned normalization.

$$\epsilon(s) = \delta \frac{1 - e^{\frac{-\Delta}{\sigma}}}{1 + e^{\frac{-\Delta}{\sigma}}} + (1 - \delta) \cdot \epsilon(s) \tag{12}$$

where the inverse sensitivity σ has a value of $\sigma = 1$ and the parameter δ that defines the relative weight of the selected action on the state-dependent exploration probability has been fixed to a value of $\delta = 1/N_q$. This last ϵ definition favors a greater degree of exploration when the difference (Δ) in Q-values before and after a learning step is large for a particular state-action pair, since that large fluctuation in the Q-value means that the policy for that state is not stable and the knowledge is uncertain. The $\epsilon(s)$ value for all the states is initialized to 1, being consistent with the fact of that at the beginning of the process the client has very little knowledge about which are good and bad actions in each particular state apart from the information provided by the initial Q-table and therefore, it needs a high ratio of exploration.

Figure 1 shows the evolution of the buffer level, the requested qualities and the available bandwidth during an interval of the simulation for the mentioned exploration policies. As can be noted the $\epsilon-$greedy method involves many sharp quality switches and several freezes because the buffer level is very unstable and it sometimes empties. On the contrary, with the other strategies the algorithm achieves a good adaptation that follows the bandwidth changes, keeps the buffer stable and close to the optimum value and avoids the occurrence of freezes.

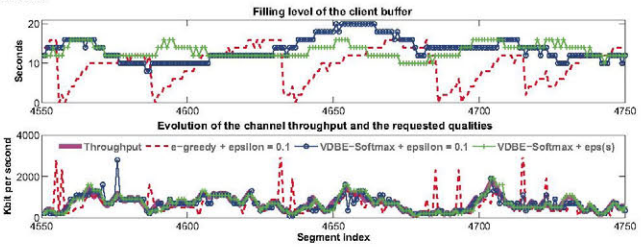

Fig. 1. QL-DASH performance with different exploration strategies.

The objective results are summarized in Table I. The ϵ-greedy method harms dramatically the QoE, since the video freezes involves a high value of the factor F. With the VDBE-Softmax method using $\epsilon = 0.1$, the action selection is quite controlled by means of the Softmax probabilities, obtaining low values for the factors that penalize the QoE: F and S. Finally, the results obtained using the VDBE-Softmax policy with the state-dependent exploration rate are very similar to the corresponding values for the previous method; in this case, the exploration is almost only performed when there is rather uncertainty regarding which is the most appropriate quality for a certain state, that is, the convergence of the algorithm on that state has not been achieved yet.

TABLE I
OBJECTIVE RESULTS FOR THE DIFFERENT EXPLORATION POLICIES

Exploration policy	QoE	Q (%)	F (%)	S (%)
ϵ-greedy, $\epsilon = 0.1$	0.667	34.58	27.71	8.86
VDBE-Softmax, $\epsilon = 0.1$	2.139	35.46	0.49	3.59
VDBE-Softmax, $\epsilon(s)$	2.135	35.41	0.49	3.73

C. Performance comparison

We have compared our proposed algorithm with Claeys' algorithm [6]. This approach is based on a method that combines eligibility traces and Q-Learning, called Watkins's Q(λ). The evaluation is performed considering two scenarios:

- In the first case, the lowest throughput equals the bitrate of the lowest content quality ($bw(1) = 100$).
- In the second case, the lowest channel throughput is lower than the lowest representation ($bw(1) = 50$), thus the possibility of freezes is very high when the channel reaches this low throughput.

Table II shows our algorithm obtains better global QoE values in both cases.

TABLE II
COMPARISON OF QOE BETWEEN THE TWO LEARNING ALGORITHMS

Algorithm	QoE	Q (%)	F (%)	S (%)
QL-DASH / bw(1) = 100	2.135	35.41	0.49	3.73
Watkins's Q(λ) / bw(1) = 100	2.038	35.62	0.81	9.5
QL-DASH / bw(1) = 50	1.301	35.25	17.17	3.72
Watkins's Q(λ) / bw(1) = 50	0.641	36.10	29.10	10.79

Looking more in detail the results, it can be observed that our algorithm obtains lower values of the F and S factors, while Claeys' algorithm achieve a slightly score in the factor Q. This means that Claeys' algorithm has selected higher bitrate segments, but this has led to the occurrence of more freezes as the value of F shows. This can be explained, since their reward function does not take into account the buffer fullness. On the contrary, our algorithm takes into account the buffer level to determine the next quality achieving a lower number of freezes and a smaller duration of them, yielding a lower factor F. Regarding the S factor, our better results are due to the inclusion of the previous requested quality as a state variable ($q(j)$), which has yield to a lower number of quality switches. On the contrary, Claeys' formulation does not consider such variable.

Finally, the significant QoE drop between the corresponding algorithms in the second scenario is due to the high number of freezes happened and their long duration, since the channel throughput of 50 Kbps is inadequate for any of the representations considered. Nevertheless, keeping the buffer level close to B_{opt}, our algorithm prevents several freezes thanks to that margin. However, the Claeys' buffer suffers several underflow situations due to its greedy behavior.

IV. CONCLUSIONS AND FUTURE WORK

We have proposed a control algorithm based on the Q-Learning approach through which the HAS client learns the optimal action in each state through its experience, evaluating the reward function and updating the Q-table. The proposed model relies on three variable states: the buffer level, the available throughput, and the previous requested segment quality. The reward function has been based on the QoE factors: the average quality, the quality switches and the playback freezes

To evaluate our solution, we have carried out some simulations comparing the performance of our approach using different methods for the selection of the exploratory actions, and we have also compared it with another self-learning HAS client. For the first evaluation, results have shown that our algorithm achieves better results with the VDBE-Softmax method whereby the action selection is based on the Softmax probabilities and in turn, they depends on the Q values. Regarding the performance comparison between the two learning algorithms, our approach has outperformed that of [6]. Our proposed model has proved its capacity to capture appropriately the dynamics of the system, and the proposed reward function has led to requesting high quality segments but taking into account the buffer level in order to avoid freezes.

In short, it is important to remark that our algorithm is inherently adaptive thanks to the exploration phase. Therefore, in case the behaviour of the system changes, it can adapt and compute optimal policies for the new environment.

Regarding the future work, we are considering more complex reward functions in order to take into account the freeze duration, since there are some cases where the occurrence of freezes is almost unavoidable but they might be minimized. In addition, we are evaluating the performance of our adaptation algorithm using subjective tests.

ACKNOWLEDGMENT

This work has been partially supported by the Ministerio de Economía y Competitividad of the Spanish Government under projects TEC2013-48453 (MR-UHDTV), ITEA2-11012 (ICARE), and IPT-2012-0306-430000 (Videocells).

REFERENCES

[1] Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats. http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html.

[2] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, "Adaptation algorithm for Adaptive Streaming over HTTP," in *Packet Video Workshop (PV), 2012 19th International*. IEEE, 2012, pp. 173–178.

[3] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *ACM MMSys 2011, Special Session: Modern Media Transport, Dynamic Adaptive Streaming over HTTP (DASH*, pp. 23–25.

[4] S. Garcia, J. Cabrera, and N. Garcia, "Quality-control algorithm for adaptive streaming services over wireless channels," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 9, no. 1, pp. 50–59, Feb 2015.

[5] J. F. T. W. W. V. L. Maxim Claeys, Steven Latré and F. D. Turck, "Design of a Q-Learning-based client quality selection algorithm for HTTP adaptive video streaming," in *Adaptive and Learning Agents Workshop, part of AAMAS2013 (ALA-2013)*, 2013, pp. 30–37.

[6] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design and optimisation of a (FA) Q-learning-based HTTP adaptive streaming client," *Connection Science*, vol. 26, pp. 25–43, 2014.

[7] C. J. C. H. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, King's College, Cambridge, UK, May 1989. [Online]. Available: http://www.cs.rhul.ac.uk/ chrisw/new_thesis.pdf

[8] A. G. Barto, *Reinforcement learning: An introduction (Adaptive Computation and Machine Learning)*. MIT press, 1998.

[9] M. Tokic and G. Palm, "Value-difference based exploration: adaptive control between epsilon-greedy and softmax," in *KI 2011: Advances in Artificial Intelligence*. Springer, 2011, pp. 335–346.